EXHIBIT 3

Interactive Communities Index

 To Table of Contents  To Interactive Entertainment  To ...

The T_Vision Project

T_Vison is an earth visualisation project. It provides a virtual globe as a multimedia interface to visualize any kind of data related to a geographic region. The virtual globe is modelled from high resolution spatial data and textured with high resolution satelite images. A T_Vision database and realtime rendering system has been developed to handle this huge amount of data. Terravison specific concept of seamless links between different levels of detail allows the continuous zooming from a global view down to recognizable features of only a few centimeters in size. On the virtual globe any kind of geographically related data can be incorporated. The user thereby has full control over which information to view, at what time and at which location.

T_Vision is a braoadbandapplication research project realised by ART+COM, finaced by DeTeBerkom, Berlin and supported by Wethernews ,Tokyo.

- T_Visions Renderer
- T_Visions Database: Terrabase
- T_Visions User Interface: The Earthtracker

- First Public Demo at ITU-Kyoto, Oct. 94
- Image Bank
- People

T_Vision is a broad-band application research project. It provides a distributed virtual globe as a multimedia interface to visualize any kind of data related to a geographic region. The virtual globe is modeled from high-resolution spatial data and textured with high-resolution satellite images.

A T_Vision database and real-time rendering system has been developed to handle this huge amount of

data. Seam-less links between different levels of detail allow contin-uous zooming from a global view down to recognizable features of only a few centi-meters in size. The virtual globe can display many types of data, including biological, sociological, economical, and others.

This project is based on the concept of a transparent and worldwide broad-band, networked topography and surface data bank. Because it is impossible for a single location to store and maintain the huge amount of high-resolution data necessary for such a visualization application, the system automatically establishes an ATM connection to the server that provides the most up-to-date and highest-resolution data required for the current field of view (and visualization layer). These remotely accessed data are integrated unobtrusively into the user's system on the fly. A special T_Vision user interface based on a large real globe ("earthtracker") facilitates the user's navigation around the virtual globe, so the user has full control over which information to view, when, and at which location.

A successful ATM T_Vision test network between Tokyo and Berlin was established in October 1994. A third node will be added this year, in Sunnyvale, California USA.

Gerd Grueneis
ART+COM e.V.
Budapesterstrasse 44
D-10787 Berlin, Germany
+49.30.254173
+49.30.25417555 fax
gruen@artcom.de
http://www.artcom.de/projects/terra/

Collaborators

Project Management
Gerd Grueneis

Ideas, Visual Database, and High Speed Networking
Pavel Mayer

Design and Concepts
Joachim Sauter

Ideas, Advanced Realtime Rendering, and Simulation
Axel Schmidt

Special Thanks
Hans Hueber, Hero Ischibashi, Joerg Jacobs, Rolf Kruse, Uli Lipka, Dirk Luesebrink, Gert Monath, Dieter Sachse, Erick Schmitz, Andreas Schneider, Christoph Stratmann, Susi Traeger, Henrik Tramberend and the rest of ART+COM.

T_Vision is funded by DeteBerkon, Berlin and supported by Wethernews Inc. Tokyo.

comments to js@artcom.de

 To Index  Tele-Garden  Virtual Discussion

The T_Vision Realtime Renderer

fly to fuji (800k mpeg)

fly to berlin (2MB mpeg)

The Task

Developping a renderer which visualize a worldwide distributed database with unlimited geometry and textures in realtime (~ 1280/1024 pixel ~ 20-30 fps).

The Platform

The IRIS Perfomer toolkit was chosen as the basic system to create the renderer. It provides the best performance especially from the multiprocessor Onyx with RealityEngine graphic but also on other machines including the future ones - and there was no need to hesitate about gl/OpenGL, quadword-aligning or things like this. Some extensions had to be added to the Performer toolkit like database paging and texture paging, because the development environment offers only 512MB main-memory and 4MB texture-memory.

The (already existing and working) Prototype

The renderer has two main parts, the render-database-manager and the render engine. The renderer database client accesses a multi-layered database of practically unlimited size. At the moment it contains surface data (satelite imagery and aerial photographs), elevation data, transparent clouds, CAD-Models of buildings and Information billboards displaying names and current temperatures of selected cities. The maximum resolution for the different data layers can be different depending on the geographical region. All datas comes from a distributed database. The renderer database client is the interface between the Performer scenegraph and the distributed earth database. On the Performer side it knows all viewing and flight parameters like position and direction. From that it calculates the currently needed data and predicts the needed data for future flightpath. Then it requests the data for a special location with an appropriate resolution. Database access by the renderer is done asynchronously without affecting the framerate.

If you approach too fast you will get a coarse image, but the frame rate is not affected. This happens only if you fly around with few 100000 km/hour below 50 km height. Data no longer needed will be removed removed from the Performer scenegraph. (if they havent been rendered in the last 30 sec or if the memory limit is reached). The database is organized as a quadtree, containg higher levels of detail as you descend down the tree.

To overcome the very limitited size of texture memory a texture pager was implemented. The difference

compared to the GL texture paging is that a particular texture is paged in before it is referenced. If the texture memory is full, the LRU tiles are freed. The paging is implemented with a callback mechanism in the drawing process.

The render engine is a full feature multiprocess Performer renderer with an still ugly graphical user interface (this will be solved in the nearer future). Beside many render options you can switch on-off the different information layers (clouds, CAD-Models, temperatures ...). You can also connect various input devices including a Earthtracker and a 6-degree-of-freedom SpaceMouse. The background color changes from black to blue when you descend into the atmosphere.

The Problems

Dynamic Performer geometry is not used, currently every patch has an absolute position on the earth. The prototype ignores the problem of the seams between different LODs, but there are several solutions for this problem which are not very computationally expensive. Another problem is that the precision of 32-Bit floats is not sufficient to represent very small features in an earth scale coordinate system. The database uses 64-Bit doubles for everything, but as you know gl can handle 32-Bit floats only. There is a solution for that, but it is not implemented yet. Currently the precision effect can be seen on objects requiring a resolution less than 1 m. This is the case specialy by entering 3D-Models of buildings. When mixing transparent layers like clouds with non transparent ones like the elevation data the NON_OCCLUDE flag don't work how exspected and there is some occlusion.

The Resources

SGI Onyx with RealityEngine2 graphics, bunch of other SGI machines, ATM network infrastructure, five years experience in realtime graphics (VR) and some experience in earth visualisation.

The T_Vision Database

Overview

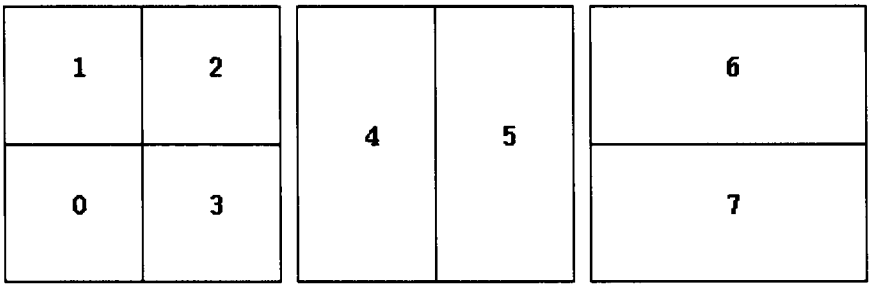
The current implementation is a very simple prototype with a very limited functionality. The database basically consists of pairs of index and data files containing 128x128 pixel texture images (surface,clouds) and 16x16 point elevation data. Geometry and Billboards are not stored at other places in the current implementation, but this will change in the future.

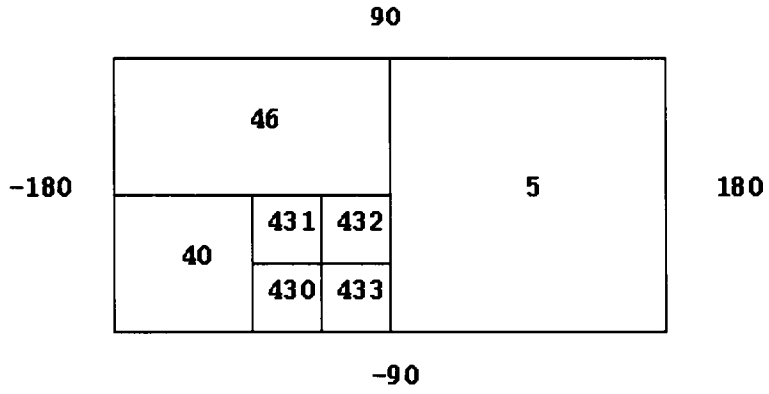
Subdividing the Planet

Main part of the key is a string we call "Global Area Identifier" (GAI). Every patch has bounds generated by binary subdivision of the whole coordinate system:

- longitude: (-180, 180) west <-> east
- latitude : (90, -90) north <-> south

There are three ways how this rectangle can be divided into 2 or 4 sectors, resulting in 8 possible sectors:





The GAI can be seen as a kind of telephone number for reaching a particular sector of the planet. The number of digits corresponds to the level of detail; the higher the number, the finer the resolution. The reason for allowing horizontal and vertical subdivision and not only the four quadrants is that if you map the rectangle onto a sphere you can choose a subdivision strategie that gives you more "quadratic"

patches, especially around the poles. Other advantages of this GAI coding compared to storing bounds:

- You dont have to worry about floating point representation inacuracies
- Fast and easy searching can be performed
- Unlimited number of subdivisions can be coded (unlimited precision)
- Codes can be used to support a very large scale data distribution on a global ATM network
- More dimensions can be added in the future

Loading the Planet

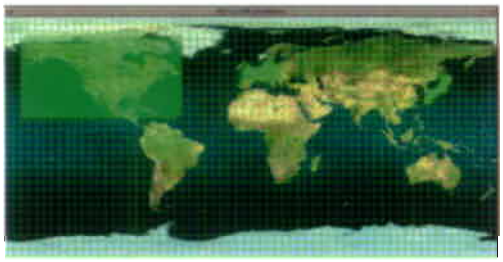
The task for the renderer database client is very simple. The renderer computes the GAIs according to the field of view and makes a simple query. In the current implementation the renderer gets a file name and an offset and performs the reading itself to let the renderer control at which timeslot this should happen. The index files are loaded at startup time and reside in the main memory. Access to remote data is done via NFS on an ATM-network, but many things this will change in the future when a "real" distributed database software will provide more powerful functionality. The performance of the current implemention allows to load 20 to 30 64k texture patches per second and is limited by I/O bandwith only.

Generating the Planet

The database is generated from various data sources like satelite imagery from LANDSAT and MOS for surface data and clouds from Meteosat, GOES and GMS weather satelites. The current generator can only use data in "cylinder projection", which means that the bounds of the map have to be parallel to the longitudes and latitudes. They can cover any part of the world with any resolution. Currently the source data consists of around 10 GB of image and DEM data, covering the whole world in 4 km/pixel, USA and Europe in 1km, Japan in 50m, some areas in USA and Germany in 50m, and parts of Berlin and Tokyo down to 30cm.

The database is generated by point sampling all patches at the required resolution for every level of detail that can be generated from the source data with magnifying the source maps. This results is a tree of different depth depending on the maximum resolution of the source maps that cover the particular sector. The generator decides for every pixel individually from which source map it is taken.

The generator can be used in a difference mode where only those patches are generated which will be different if you add a new source map. This is necessary to avoid generating the whole database after adding a new map of a small region like the center of Berlin, which does only affect a few existing patches in the database.
A simple graphical user interface exists to browse through the database and to view and position the source data:



Problems with the Planet

If there is one thing about the earth that I learned then this is the fact that the earth is really, really big. Bigger than your main memory, bigger than your filesystem, bigger than a 32Bit address space, and bigger than the precision of a 32Bit float.
Unfortunetaly, with IRIX 5.3 a lot of work had to go into working around system limitations, for example a "virtual" virtual memory manager that lets you map dozen of gigabytes into your 4GB (in fact 2GB!) of virtual memory. Even worse, with RE2 graphics subsystem you can not simply store your coordinates in 32 Bit floats, but you have to scale and translate everything(!) while flying around, your camera, your speed, your model, your clipping planes. The database generator uses 64Bit doubles everywhere, but the renderer can not take full advantage of that at the moment.

The Future of the Planet

One of the first things I will do is to throw away all the code I have written for this prototype and replace it with a real object oriented full feature distributed high performance real time database. How I get it ? I will buy one, if possible, or we will build something around an existing database kernel, or use some sophisticated tools to let the computer do most of the programming. :-)

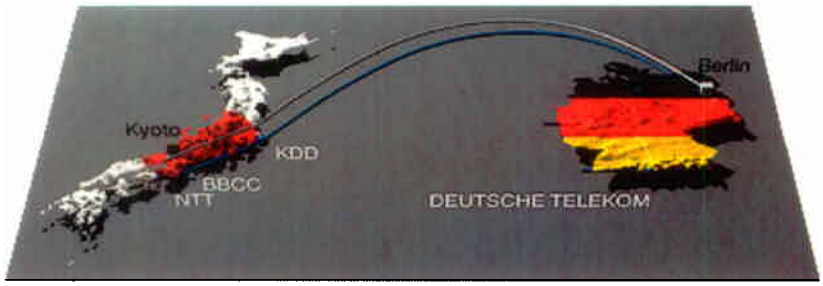
Seriously, there is a real need for some new operating systems, distributed object oriented real time operating systems.

Meanwhile we will work with the best stuff we can get.

The TERRAVISION User Interface: The Earthtracker

A combination of two interfaces is used to navigate around the virtual globe:
A Space Mouse is used to control the viewers position and angle of view. And in order to move the Earth itself a so-called Earthtracker, a special terravision user interface in form of a large real globe was developed. movie (4 MB)

T_Visions First Public Demo at ITU-Kyoto, Oct. 94



In 1994, government representatives of the 130 member states of the International Telecommunications Union met in the Japanese city of Kyoto - to consider future strategies and perspectives for telecommunication, and to present the latest developments. At the invitation of the japanese Ministry of Post and Telecommunications, applications for the future broadband networks, such as B-ISDN, were demonstrated in an accompanying exhibition. This exhibition featured the opening of the Data Highway between Germany and Japan - with German Telekom, NTT and KDD combining their Broadband networks over a common ATM connection. T_Vision was publicly presented for the first time as part of the virtual Ribbon Cutting ceremony for the new Infobahn.

TERRAVISION Image Bank

This is free press material for noncomercial purposes. To get a key for ftp the high resolution images please send an e-mail for to js@artcom.de

terra_globe.tiff, 897 x 895 Pixel, RGB, 806 kByte

terra_fuji.tiff, 1185 x 987 Pixel, RGB, 2.728 kByte

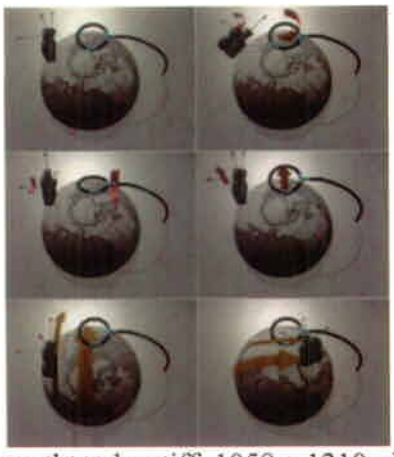
terra_fly1.tiff, 1186 x 986 Pixel, RGB, 2.200 kByte


terra_fly2.tiff, 1186 x 986 Pixel, RGB, 2.410 kByte


terra_fly3.tiff, 1186 x 986 Pixel, RGB, 3.040 kByte


terra_fly4.tiff>, 1186 x 986 Pixel, RGB, 2.040 kByte


>earthtrackerblau.tiff, 658 x 486 Pixel, RGB, 420 kByte



earthtracker.tiff, 1058 x 1210 pixel, RGB, 960 kByte

People

Gerd Grüneis, *Project Management;*
Pavel Mayer, *Ideas, Visual Database & High Speed Networking;*
Joachim Sauter, *Design and Concepts;*
Axel Schmidt, *Ideas, Advanced Realtime Rendering and Simulation.*

special thanks:
Hans Huebner, Hero Ischibaschi, Joerg Jacobs, Rolf Kruse, Uli Lipka, Gert Monath, Dieter Sachse, Andreas Schneider, Christoph Stratmann, Susi Traeger, Henrik Tramberend and the rest of ART+COM.